Date 20/07/21

Neeraj Appan A.I

Practical-03

* Aim- Implement A* search algorithm for romanian map problem

* A* search Algorithm Description-

-> A* Search algorithm is an informed search algorithm or a best first search, meaning that is formulated weighted graphs; starting from a specific starting node of a graph; it aims to find a path of the given goal node having the smallest cost

* Strategies evaluated along the following dimensions

1) Completeness - does it always find a solution if one exits?

-> The A* algorithm is the optimal algorithm provided heuristic function is underestimated, complete and admissible

2) Optimality - does it always find a least cost solution

-> A search algorithm is optimal if no other search algorithm uses less time or space or expands fewer nodes, both will guarantee of solution quality. The optimal search algorithm would be on that picks the correct node at choice

3) Time complexity - number of node generated

-> The time complexity of A* depends on heuristic. In worst case of an unbounded search space, the number of nodes expanded is exponential in the depth of solution

Neeraj Appan ~~Appu~~

SDS Page No.
Date / /

4) Space complexity - maximum number of nodes in memory O(|V|) = O(b^d), Stores all generated nodes in memory
5) Time and space complexity are measured in terms of
a) b = maximum branching factor of search tree = O(b^d)
b) d = depth of least cost solution is Shortest path (∞)
c) m = maximum depth of state space (may) (∞)
d) search cost (time), total cost (time + space)

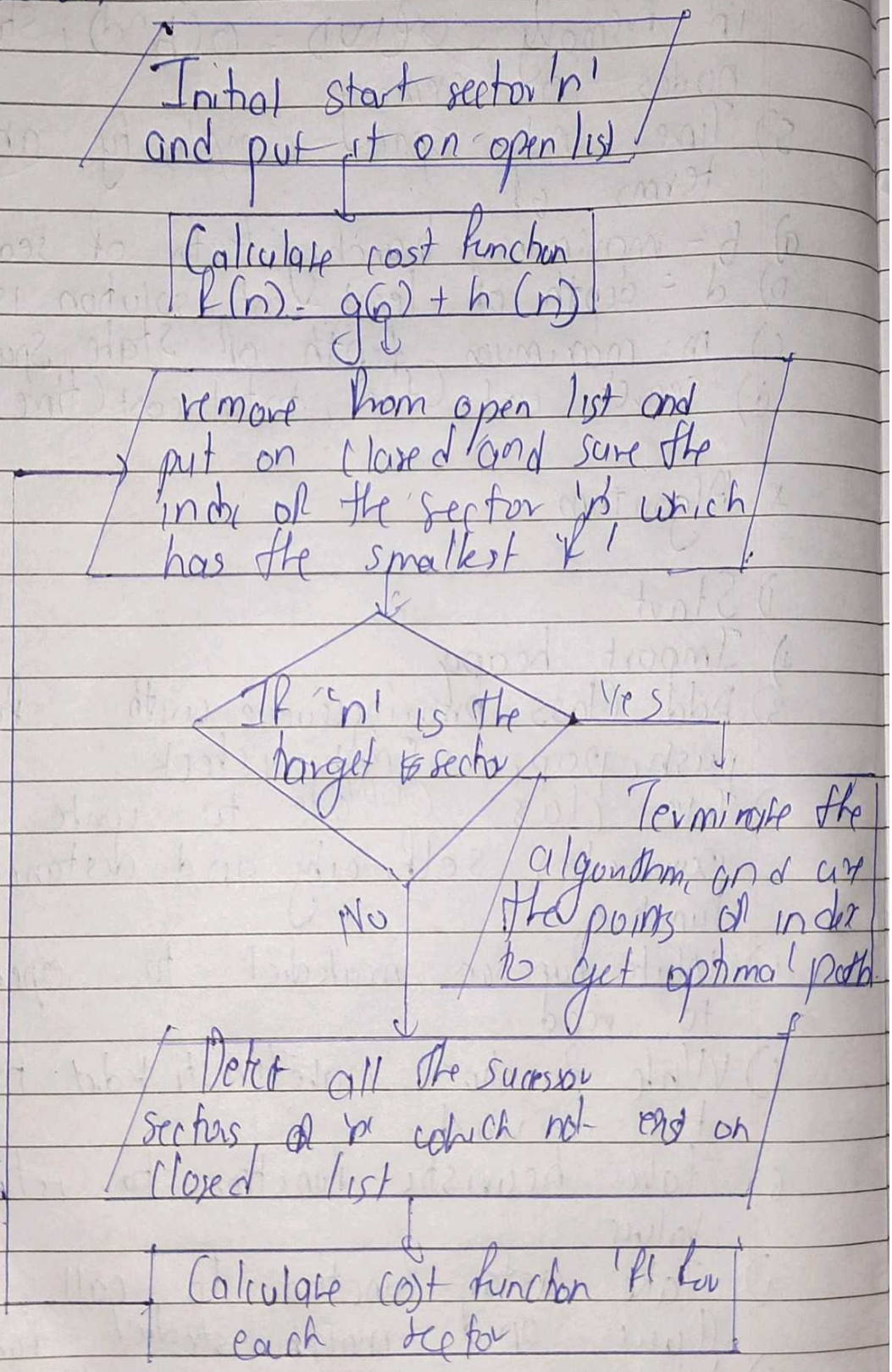
* Algorithm

1) Start
2) Import heapq
3) Add Class priorityQueue with functions - init - push, pop, is Empty, check
3) Take class City to write parameters self, city and distance in - init - function
4) Write function makedict to open romania.txt to read
5) Write function makehuristikdict to open romaniaSL.txt to read
6) Take heuristic function to return nodes and values
7) Take astar function to call class priority Queue and makehuristdict function
8) Write parameteres necessary to print output in printoutput class
9) Give source and dishnation in main function
10) Stop

Neeraj Appari

SDS | Page No.
Date / /

* Flowchart



try 3.py - E:/fffiiles/college pracs and projects/AI/try 3.py (3.8.3)

File Edit Format Run Options Window Help

```
import heapq
print("Neeraj Appari")


class priorityQueue:
    def __init__(self):
        self.cities = []

    def push(self, city, cost):
        heapq.heappush(self.cities, (cost, city))

    def pop(self):
        return heapq.heappop(self.cities)[1]

    def isEmpty(self):
        if (self.cities == []):
            return True
        else:
            return False

    def check(self):
        print(self.cities)


class ctNode:
    def __init__(self, city, distance):
        self.city = str(city)
        self.distance = str(distance)


romania = {}
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: E:/fffiiles/college pracs and projects/AI/try 3.py =========
Neeraj Appari
Program algoritma Astar untuk masalah Romania
        Arad => Bucharest
============================================================
Kota yg mungkin dijelajah               : ['Arad', 'Sibiu', 'Rimnicu Vilcea', 'Fagaras', 'Pitesti', 'Bucharest']
Jumlah kemungkinan kota                 : 6
============================================================
Kota yg dilewati dg jarak terpendek     : ['Arad', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest']
Jumlah kota yang dilewati                         : 5
Total jarak
: 418
>>>
```

Type here to search

ENG 09:21 20-07-2021

try 3.py - E:/fffiiles/college pracs and projects/AI/try 3.py (3.8.3)

File Edit Format Run Options Window Help

```
    distance = {}
    q = priorityQueue()
    h = makehuristikdict()

    q.push(start, 0)
    distance[start] = 0
    path[start] = None
    expandedList = []

    while (q.isEmpty() == False):
        current = q.pop()
        expandedList.append(current)

        if (current == end):
            break

        for new in romania[current]:
            g_cost = distance[current] + int(new.distance)

            # print(new.city, new.distance, "now : " + str(distance[current]), g_cost)

            if (new.city not in distance or g_cost < distance[new.city]):
                distance[new.city] = g_cost
                f_cost = g_cost + heuristic(new.city, h)
                q.push(new.city, f_cost)
                path[new.city] = current

    printoutput(start, end, path, distance, expandedList)


def printoutput(start, end, path, distance, expandedlist):
    finalpath = []
```

Ln: 93 Col: 0

try 3.py - E:/fffiiles/college pracs and projects/AI/try 3.py (3.8.3)

File Edit Format Run Options Window Help

```
romania = {}


def makedict():
    file = open("romania.txt", 'r')
    for string in file:
        line = string.split(',')
        ct1 = line[0]
        ct2 = line[1]
        dist = int(line[2])
        romania.setdefault(ct1, []).append(ctNode(ct2, dist))
        romania.setdefault(ct2, []).append(ctNode(ct1, dist))


def makehuristikdict():
    h = {}
    with open("romania_sld.txt", 'r') as file:
        for line in file:
            line = line.strip().split(",")
            node = line[0].strip()
            sld = int(line[1].strip())
            h[node] = sld
    return h


def heuristic(node, values):
    return values[node]


def astar(start, end):
    path = {}
```

Ln: 61 Col: 0

try 3.py - E:/fffiiles/college pracs and projects/AI/try 3.py (3.8.3)

File Edit Format Run Options Window Help

```
def printoutput(start, end, path, distance, expandedlist):
    finalpath = []
    i = end

    while (path.get(i) != None):
        finalpath.append(i)
        i = path[i]
    finalpath.append(start)
    finalpath.reverse()
    print("Program algoritma Astar untuk masalah Romania")
    print("\tArad => Bucharest")
    print("=========================================================")
    print("Kota yg mungkin dijelajah \t\t: " + str(expandedlist))
    print("Jumlah kemungkinan kota \t\t: " + str(len(expandedlist)))
    print("=========================================================")
    print("Kota yg dilewati dg jarak terpendek\t: " + str(finalpath))
    print("Jumlah kota yang dilewati \t\t\t: " + str(len(finalpath)))
    print("Total jarak \t\t\t\t\t: " + str(distance[end]))


def main():
    src = "Arad"
    dst = "Bucharest"
    makedict()
    astar(src, dst)


if __name__ == "__main__":
    main()
```

Ln: 121 Col: 0

romania - Notepad

File Edit Format View Help

```
Arad,Zerind, 75
Arad,Sibiu, 140
Arad,Timisoara, 118
Zerind,Oradea, 71
Oradea,Sibiu, 151
Timisoara,Lugoj, 111
Sibiu,Fagaras, 99
Sibiu,Rimnicu Vilcea, 80
Lugoj,Mehadia, 70
Fagaras,Bucharest, 211
Rimnicu Vilcea,Pitesti, 97
Rimnicu Vilcea,Craiova, 146
Mehadia,Dobreta, 75
Bucharest,Pitesti, 101
Bucharest,Urziceni, 85
Bucharest,Giurglu, 90
Pitesti,Craiova, 138
Craiova,Dobreta, 120
Urziceni,Hirsova, 98
Urziceni,Vaslui, 142
Hirsova,Eforie, 86
Vaslui,Lasi, 92
Lasi,Neamt, 87
```

Windows (CRLF) Ln 1, Col 1 100%

Type here to search

ENG 11:16 20-07-2021

romania_sld - Notepad

File Edit Format View Help

```
Arad, 366
Bucharest, 0
Craiova, 160
Dobreta, 242
Eforie, 161
Fagaras, 176
Giurgiu, 77
Hirsowa, 151
Lasi, 226
Lugoj, 244
Mehadia, 241
Neamt, 234
Oradea, 380
Pitesti, 100
Rimnicu Vilcea, 193
Sibiu, 253
Timisoara, 329
Urziceni, 80
Vaslui, 199
Zerind, 374
```

Windows (CRLF) Ln 1, Col 1 100%

Type here to search

ENG 11:16 20-07-2021

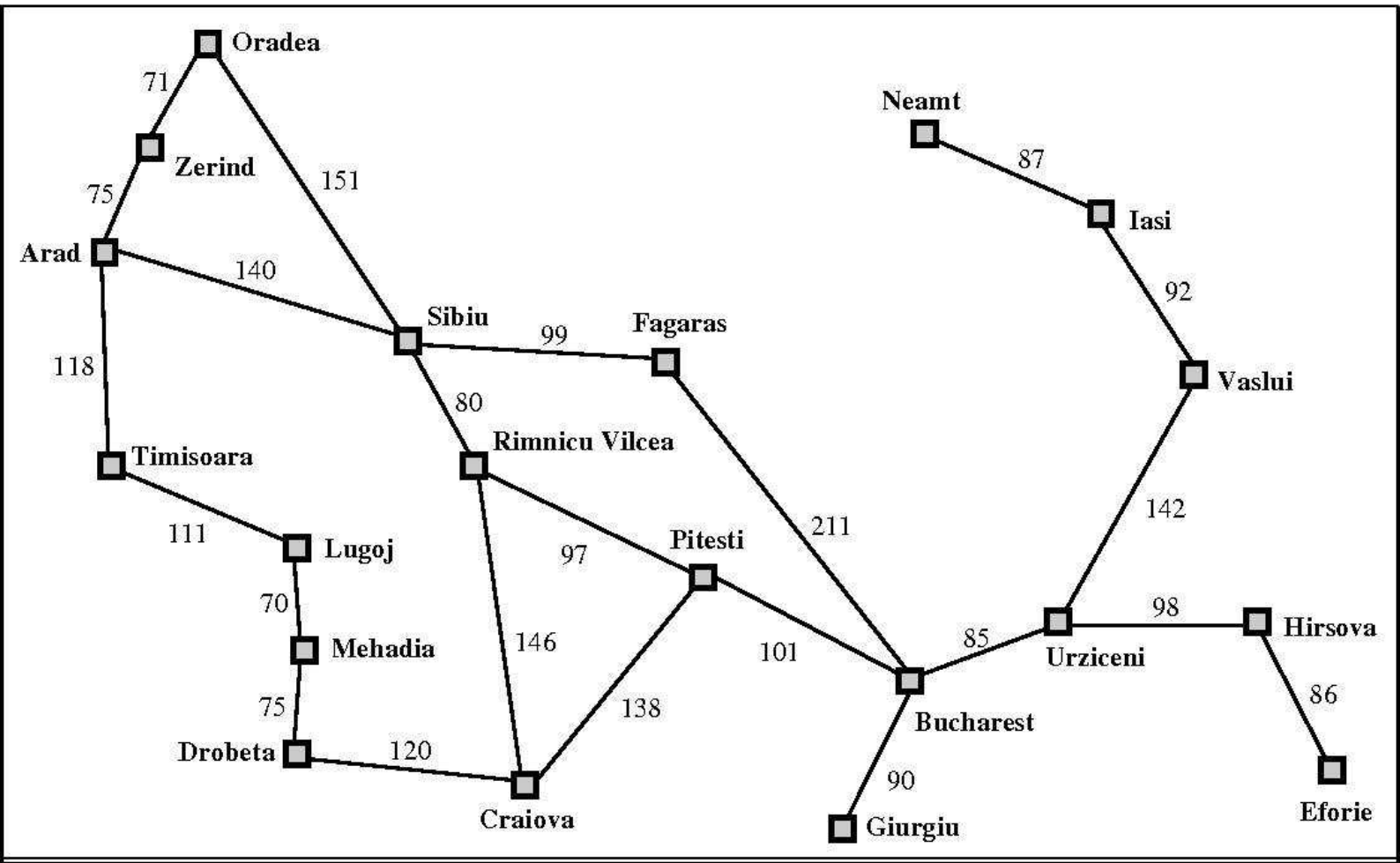


Figure 3.2 A simplified road map of part of Romania.